



**I denne artikel om COMAL 80 / Z80 ser vi nærmere på nogle af de kommandoer, der gør Comal til et så stærkt og overskueligt sprog, som det er.**

Betingede sætninger er, ligesom de løkker vi gennemgik i sidste nummer af bladet, en vigtig del af et programmeringssprog, og Comal er udstyret med muligheder for både løkker og betingelser.

Comal's IF-sætning findes i mange udgaver:

Den mest simple hedder: IF  $a=b$  THEN  $c:=d$ , og den behøver næsten ikke forklaring, men alligevel: Hvis  $a$  er lig med  $b$  sættes  $c$  lig med  $d$  og ellers går programmet blot videre til næste linie. Desuden er det mange gange flere ting der skal udføres, hvis en special betingelse er sand, og så er den simple udgave ikke nok. Her er man nødt til at anvende IF-ENDIF sætningen, f.eks.:

```
IF a=b THEN
  c:=d
ENDIF
```

Her bliver alt hvad der ligger mellem IF og ENDF udført, hvis betingelsen er sand, dvs.  $a$  er lig med  $b$ . Længere mærker til at alle tilføjelser kan skrives på en linie adskilt med semikolon, f.eks.  $c:=d; e:=1$  osv. Dog, at vi i ovenstående program eksempel godt kunne have skrevet IF  $a=b$  THEN  $c:=d; e:=1$ .

Alle Comal's konstrukter kan blandes ind i hinanden fuldstændig smidigt, bare man sørger for at ENDF passer til IF, UNTIL or REPEAT osv., altså:

```
IF a=b THEN
  REPEAT
    c:=1
  UNTIL c=a
ENDIF
```

Det går altså ikke at blande sætningerne vilkårligt og så bliver et Comal'en selv finder ud af det. Prøv omgang følgende eksempel:

```
IF a=b THEN
  REPEAT
    c:=1
  UNTIL c=a
ENDIF
```

COMAL-80 fortæller dig med det samme, at den er gal, hvis programmet starter med REPEAT.

Men tilbage til IF. Oftest skal programmet gøre en ting, hvis betingelsen er sand, og en anden hvis den er falsk. Derfor kan man bruge en IF-ELSE-ENDIF sætning:

```
IF a=b THEN
  c:=d
ELSE
  e:=f
ENDIF
```

Det fremgår tydeligt, at programmets struktur, hvordan det ene og det andet udføres. Det er en af de ting, der gør

det behageligt at programmere i COMAL - det er på den måde at følge et forløb i andres programmer. Man kan selvfølgelig også gøre det mere indviklet:

```
IF a=b THEN
  IF a>b THEN
    c:=d
  ELSE
    d:=e
  ENDF
ELSE
  e:=f
ENDIF
```

Det fremgår stadig tydeligt, hvordan de forskellige sætninger hænger sammen, og det er faktisk mere overkommeligt at forklare det, men jeg prøver alligevel.

Hvis  $a$  er lig med  $b$  og  $a$  større end  $b$  sættes  $c$  lig med  $d$  ellers hvis  $a$  mindre eller lig med  $b$ , sættes  $d$  lig  $e$  og hvis  $a$  er forskellig fra  $b$  sættes  $e$  lig med  $f$ . Så simpelt er det - Ther's all, Folks!

Man kan lave IF-sætninger i det sandeligste, men på et punkt er de helt ueverlige. Hvis man skal sammenligne een variabel med en række forskellige muligheder kan IF-sætningen komme til at se således ud:

```
IF a=1 THEN
  ET
ELSE
  IF a=2 THEN
    TO
  ELSE
    IF a=3 THEN
      FIRE
    ... (ovs) ...
  ENDF
ENDIF
ENDIF
ENDIF
```

Som du ser, hvis der bare er 10 mulige IF-sætninger, bliver det rimelige overkommeligt. En ting der kan hjælpe lidt er ELIF-sætningen - en ELSE og en IF sætning, bygget sammen:

```
IF a=1 THEN
  ET
ELIF a=2 THEN
  TO
ELIF a=3 THEN
  FIRE
ELIF a=4 THEN
  ... (ovs) ...
ENDIF
```

Men det er altså arbejde at skrive alle ELIF-sætninger.

net, når de er afsluttes. Derfor har man i COMAL lavet en CASE-struktur, og den kan laves, se således ud:

```
CASE a OF
  WHEN 1
  ET
  WHEN 2
  TO
  WHEN 3
  THE
  WHEN 4
  FIRE
  ... ( osv. ) ...
ENDCASE
```

CASE-sætningen søger alle WHEN sætninger igennem, indtil værdien af variabelen a findes, herefter den sætning det, der står efter WHEN. Dette fortsættes indtil næste WHEN. Hvis COMAL når til ENDCASE, og der ikke er en WHEN der passer til CASE-sætningen, udskrives en fejl-melding. Dette kan undgås ved at bruge OTHERWISE, som bl.a. benyttes når man ikke er helt sikker på at CASE-sætningen passer med en WHEN.

```
CASE a OF
  WHEN 1
  ET
```

```
  // Nu skal instruktionen jo ikke tro jeg kommer let til
  // heromst ved at skrive alle de korte linier
  OTHERWISE
```

FEIL // Udløst kan, hvis en rigtig WHEN-sætning mangler.

```
ENDCASE
```

BASIC har nogenlunde de samme faciliteter som IF-THEN-ELSE-og ON-CASE sætningerne, men de har (indtænkt) lidt deres mangler. BASIC er i realiteten et Håsters sprog. Hvis vi et eksempel skulle oversættelse til BASIC, ville linier komme til at se således ud:

```
100 IF a=1 THEN GOSUB 110 ELSE IF a=2 THEN
GOSUB 200 ELSE IF a=3 THEN GOSUB 300 ELSE IF
a=4 THEN GOSUB 2322
```

hvilken vi ikke kan kalde særligt overskueligt at følge. Vi er endda heldige i Amstrad-BASIC, da ELSE ikke findes i mange BASIC-versioner.

Særligt er ON-et eller andet-GOTO/GOSUB udtrykket bedre i dette specielle tilfælde, da variabelen a har en konstant værdi fra 1 til 4, dvs. 10, så linien kunne skrives således: ON a GOSUB 110, 200, 300, 2322, ... men hvis a ikke var konstant, men i stedet en blanding af forskellige tal, var COMAL alligevel bedst ud - selv til multichoice (flere mulige valg på samme linie).

```
CASE a OF
  WHEN 1,2,3
  EN-TO-TRE
  WHEN 4,5,6
  MEN-TO-TRE
  WHEN 0
```

```
NUL-SLUT-PUNKTUM-FINALE
OTHERWISE
FARVEL-SES-OM 8 UGER
ENDCASE
```

Og der var alt for nu. - LAR

## Comal-80 i Tyskland

Som formidler af COMAL-80 kan jeg nævne at et tysk firma ved navn COMALGRUPPE Deutschland er begyndt

at markedsføre COMAL-80/280 rev. 1.82 i tysksprogede lande. Denne COMAL version har tyske fejlmeldelser, og sælges med 5 forskellige markedspriser: GRAFER (TURTLE), JOYSTICK, DEKEY, m.fl. Prisen ligger på 69,- DM, dvs. ca. 250 kr.

Det skal understreges, at firmaet IKKE har tilladelse til at sælge COMAL-80 versionen til Danmark eller andre lande, der ikke har tysk som hovedsprog.

## Reserveret hukommelse

Som de fleste jo nok har fundet ud af, kan man med NEW kommandoen reservere et antal bytes, som man eventuelt kan bruge til at lagre maskinkode-rutiner eller andet i. Der er flere, der har klaget over at indholdet af dette område IKKE gemmes sammen med programmet ved SAVE. (Da det er et større arbejde at lave dette, vil jeg hellere i første omgang nævne med et forklare, hvordan man dynamisk kan reservere hukommelse:

Hvis man i en linie har følgende sætning:  
0020 DIM dummy8-OF 367

bliver der altså 367 bytes til værdien af dummy8 et eller andet sted i hukommelsen, det er nu vores job at finde dette område. I linien umiddelbart efter DIM-sætningen, skal følgende sætning stå:

```
0020 address = DPEDD45302) + 4
```

Hvilket betyder at variabelen address' ridses adressen, lever de 367 bytes ligger. Disse linier kan naturligvis også stå i en faldet procedure, i så fald vil de reserverede bytes være tilgængelige for den pågældende procedure.

Fredrik Kristensen



PROLOG SPECIALISTEN  
leverer selvfølgelig også til AMSTRAD.

## PROLOG

er fremtidens programmeringssprog  
der kan gøre de AMSTRAD  
memorabelt og intelligent.

## AMSTRAD

fortjener det bedste programmeringssprog  
der kan købes.

PROLOG til AMSTRAD fra

((PROLOG)) ((DATA))  
Myntvej 3, 1086 Vestbjerg  
tlf. 08 - 55 61 24